Martin Eisemann; Marcus Magnor

Filtered Blending and Floating Textures: Ghosting-free Projective Texturing with Multiple Images

Auch erschienen als:
Technical Report 2007-5-3. - Computer Graphics Lab, TU Braunschweig

Dipl. Inf. Martin Eisemann
eisemann@cg.tu-bs.de

Prof. Dr. Ing. Marcus Magnor
magnor@cg.tu-bs.de
Computer Graphics Lab, TU Braunschweig

Filtered Blending and Floating Textures: Ghosting-free Projective Texturing with Multiple Images

Technical Report 2007-5-3

May 23, 2007

Computer Graphics Lab, TU Braunschweig

Abstract

Whenever approximate 3D geometry is projectively texture-mapped from different directions simultaneously, annoyingly visible aliasing artifacts are the result. To prevent such ghosting in projective texturing and image-based rendering, we propose two different GPU-based rendering strategies: filtered blending and floating textures. Either approach is able to cope with imprecise 3D geometry as well as inexact camera calibration. Ghosting artifacts are effectively eliminated at real-time rendering frame rates on standard graphics hardware. With the proposed rendering techniques, better-quality rendering results are obtained from fewer images, coarser 3D geometry, and less accurately calibrated images.

Contents

Chapter 1

Introduction

Figure 1.1: Images from our test data sets: for the synthetic *Bunny* and the real-world captured *Garfield*, approximate 3D geometry models are available. For the synthetic light fields *Buddha* and *Dragon*, a planar surface must suffice as geometry proxy. See Tab. 1.1 for more information on our test data sets.

Approximate geometry, camera calibration inaccuracies, and subcritical sampling are the causes for ghosting artifacts in light field rendering [LH96], lumigraph rendering [GGSC96], and view-dependent projective texture mapping [DYB98]. In fact, ghosting/aliasing/double images, throughout the paper, we use the terms "ghosting", "double images" and

	Bunny	**Garfield**	**Buddha**	**Dragon**
# geometry primitives	948	1280	1	1
Total images	49	24	256	256
Pixels per image	512^2	768×576	256^2	256^2
Object size (width, height, depth)	(1.0, 0.98, 0.76)	(0.14, 0.16, 0.14)	(2.0, 2.0, 0.0)	(2.0, 2.0, 0.0)
Set depth uncertainty	0.01	0.003	0.2	0.23
Band-limit filter support	12 pixels	10 pixels	12 pixels	10 pixels
Viewport	$360° \times 360°$	$360° \times 180°$	$90° \times 90°$	$90° \times 90°$
Output resolution (pixels)	512^2	512^2	512^2	512^2
Type	synthetic	real-world	synthetic	synthetic

Table 1.1: Information concerning our test data sets shown in Fig. 1.1.

"aliasing" synonymously, is a problem common to all image-based modeling and rendering applications whenever recorded image footage is to be merged with available geometry into one consistent representation. Already small imprecisions in camera calibration reduce attainable 3D reconstruction accuracy. Inaccurately calibrated images as well as approximate 3D geometry result in inconsistencies during registration which, subsequently, cause aliasing during rendering. For highly accurate, laser-scanned geometry, a number of strategies have been devised how to register photographs to 3D geometry in the presence of calibration inaccuracies [WAA+00, BMR01, LKG+03], as well as how to generate a globally consistent texture map from multi-view footage [RCMS99, Bau02, ZWT+05]. While impressive digital models of real-world objects have been created this way, they come at the price of considerable user interaction, and not all approaches are suitable for reproducing view-dependent reflectance effects.

In this paper, we present two different algorithms to achieve aliasing-free rendering results directly from a set of photographs in conjunction with some arbitrarily coarse geometry proxy. The set of images may only be approximately calibrated, images and geometry proxy may not be accurately registered, and the geometry proxy may be as inexact as a planar surface, e.g. in light field rendering. Based on the notion of projective texture-mapping, both approaches we propose eliminate ghosting artifacts on-the-fly during rendering on the GPU. Our first strategy, *filtered blending*, adaptively low-pass filters the images used for projective texture-mapping depending on the current viewpoint and geometry inaccuracy. Varying with the viewpoint, the texture images are individually blurred just enough to prevent ghosting on the surface. In contrast, *floating textures* locally slide on the geometry surface to match up, driven by the optical flow field between the projected images. This way, high-frequency image details are preserved even if only a very coarse geometry proxy is available.

Our goal is to improve the visual quality of existing image-based modeling and rendering methods as well as to simplify the use of image-based approaches for representing real-world objects. As particular contributions, our paper presents

- a view-dependent, anisotropic reconstruction filter that is able to take camera sampling density into account, and
- a novel texturing algorithm that constitutes a symbiosis between classical linear interpolation and optical flow-based warping refinement.

Both approaches yield visibly improved rendering results over conventional multi-image texturing and allow for real-time frame rates.

Our paper is organized as follows. After reviewing relevant previous work in Sect. 2 we examine the underlying problem of ghosting artifacts in multi-image projective texture mapping, Sect. 3. In Sect. 4 we describe filtered

blending and floating textures as two different ways to eliminate ghosting. Implementation details are given in Sect. 5, and experimental evaluation results are presented in Sect. 6 before we conclude with Sect. 7.

Chapter 2

Related Work

Image-based rendering (IBR) methods are able to achieve highly realistic rendering results of real-world objects or scenes from a collection of calibrated photographs. While some IBR methods rely solely on image number to minimize aliasing artifacts [LH96, MP04], most IBR approaches make additional use of scene depth [GGSC96, IMG00, BBM+01, ZKU+04], or full 3D geometry [DYB98, CTMS03, VBK05, SSS06]. All IBR techniques require accurate camera calibration during acquisition. Additional scene depth or 3D geometry information is either reconstructed directly from the set of calibrated images, modeled by hand, or measured independently. Potential sources for aliasing artifacts during rendering are (1) image calibration inaccuracies, (2) subcritical sampling in conjunction with insufficient prefiltering [CCST00, LS04], and possibly (3) imprecise depth maps or inexact geometry.

Image-based modeling (IBM) extends the notion of IBR in that high-quality 3D geometry scans of an object are augmented with a collection of photos to capture its visual appearance [RCMS99, WAA+00, Bau02, LKG+03, ZWT+05]. To register the images with the 3D model, accurate camera calibration is necessary, and a suitable model surface parameterization must be available to map the images to texture domain. Finite scanner resolution and tolerances, registration inaccuracies, and camera calibration errors all degrade overall image-to-texture mapping accuracy.

Different reconstruction filters for IBR have been investigated in the literature. Based on an analysis of the sampling problem in frequency and geometry space, respectively, by Chai *et al.* [CCST00] and Lin *et al.* [LS04], one can apply a low-pass filtering to the input/output images for ghosting-free "band-limited reconstruction" [SYGM03]. Since the filtering operation is performed as a pre-processing step based on the maximum disparity, the current viewing position cannot be taken into account. Hence, the rendering result is excessively blurred and more-than-needed image detail is lost. The band-limiting approach is also further complicated if cameras

are non-uniformly distributed [DTM96]. Isaksen *et al.* [IMG00] propose a "wide-aperture reconstruction filter" which increases the spatial support or aperture size of the reconstruction filter. In theory, this allows reconstructing any individual scene element without ghosting artifacts. Unfortunately, different scene elements at different distances from the focal plane are excessively blurred, and view-dependent reflectance characteristics are lost. A combination of the two approaches is proposed by Stewart *et al.* [SYGM03]. A quadralinear reconstruction is applied and the resulting image is low-pass filtered, maintaining some of the view-dependent information, while higher frequencies are added back in from the sharply focused features created by the wide-aperture reconstruction filter [IMG00]. Alternatively, Liu *et al.* [LCM+06] estimate scene geometry dynamically using a color similarity-based plane sweeping algorithm. While ghosting artifacts are reduced in the final image, new artifacts are introduced due to random color similarities. With wider camera baselines, these mismatch artifacts increase disproportionately.

Optical flow techniques provide important tools in motion analysis and are applicable to problems in IBR as well. Since the seminal papers of Horn and Schunck [HS81] and Lucas and Kanade [LK81] a variety of optical flow approaches have been developed. Fast optical flow reconstruction algorithms are, in general, limited in the maximum distance between corresponding image features which typically may not lie farther apart than a handful of pixels.

Chapter 3

Problem Description

In this section, we take a closer look at the causes of ghosting in projective texture mapping and light field rendering. We show that ghosting is solely dependent on the maximum disparity of a projected scene point to its real position in texture space. Therefore, ghosting can be detected even if only a single input camera and the virtual camera is taken into account.

Every pixel of the input images can be seen as the weighted integral of the light arriving at the image plane of the camera. Every scene point L of sufficiently small size, compared to the camera resolution, therefore contributes exactly to one pixel in the recorded images. During rendering, these are then reprojected onto an approximated surface. Lin *et al.* [LS04] state that the intensity contributions of each scene point L must at least touch each other in the output image to avoid ghosting. This is true for light field approaches if virtual and capturing cameras have the same resolution and the user is restricted to stay outside the convex hull defined by camera and focal plane. However, other rendering approaches, like projective texturing and IBR, demand a more general definition of ghosting. Thus we propose that every scene point L must provide a single, resolution independent intensity maximum in the output image. Note that this definition reduces to the one proposed by Lin *et al.* , if the above preconditions are fulfilled.

As one example, let's consider the viewing ray C_vL as shown in Fig. 3.1. Projecting the input image of camera C_1 onto the focal plane/approximate surface, the contribution of L will not appear at point F_0, but at point F_1, revealing a disparity of d. If d, projected into the output and input image, is larger than one pixel in both of them, ghosting artifacts may appear.

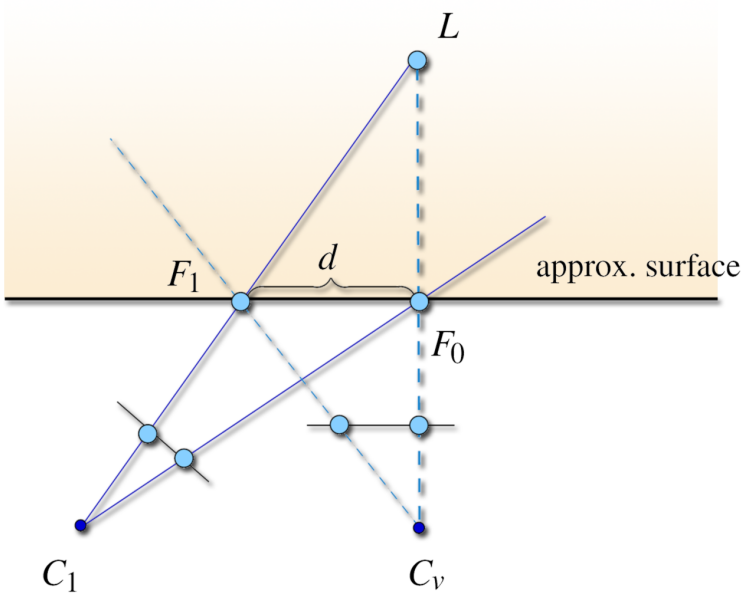


Figure 3.1: Ghosting in projective texture mapping, view-dependent texture mapping and light field rendering: The actual scene point L, as recorded from cameras C_1 and C_v, is projected to two different points F_1 and F_0 on the approximate object surface. If the distance $d = F_0 - F_1$ is larger than one pixel in the input/output images, ghosting occurs.

Chapter 4

Aliasing-free Projective Texture Mapping

Essentially there are three already established ways to circumvent the problem of double images. The first approach is to reduce the sample spacing between the cameras during recording to remove the undersampling and is the usual approach in light field rendering [LH96]. This method is not possible though if the input images are already given and memory consumptions can become quite high. The second method is to increase the accuracy of the underlying geometry [GGSC96, DTM96], and is usually not possible or wanted. And finally one could apply a more appropriate low-pass or decimation filter to the plenoptic function [CCST00, IMG00, LS04, SYGM03]. This can be done either as a preprocess on the input images, the resulting image or, as we will do in our filtered blending approach, in texture space seperately for every fragment and input image, depending on the viewing-position. Additionally we propose a novel way to avoid ghosting by our floating textures technique. The idea is to deal with the inaccuracies in the input samples directly, before reconstructing the plenoptic function [AB91], using an iterative approach.

4.1 Filtered Blending for View-Dependent Projective Texture Mapping

To motivate our first rendering approach, consider the diagram in Fig. 4.1. The scene point L lies on the line of sight of the viewing ray $C_v L_{p_0}$, somewhere within the interval of maximum depth uncertainty d_{max} from the approximate geometry. The line segment $L_{p_1} L_{p_2}$ projected into the texture space of C_1 reveals another line segment $T_{L_{p_1}} T_{L_{p_2}}$, which we call the line of disparity. Any value on this line could be the correct texture value. This is in fact similar to an epipolar geometry constraint [HZ03].

We solve this uncertainty problem in a resampling process. Choosing

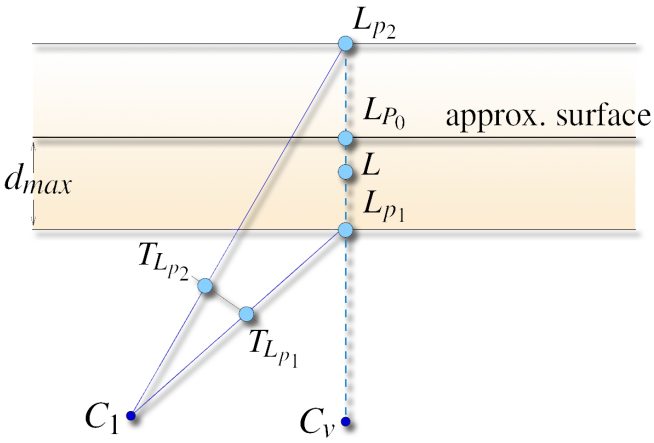


Figure 4.1: Scene point estimation. The scene point L observed by the viewing ray $C_v L_{p_0}$ can only be estimated to lie somewhere between L_{p_1} and L_{p_2}, which are defined by the maximum depth uncertainty d_{max} from the approximate surface. Its correct color value observed by camera C_1 lies somewhere between the projected texture coordinates $T_{L_{p_1}}$ and $T_{L_{p_2}}$.

$s_p = \frac{1}{2} T_{L_{p_1}} T_{L_{p_2}}$ as the sampling position and $s_d = |T_{L_{p_1}} T_{L_{p_2}}|$ as the sampling distance, we anisotropically resample the texture function along the line of disparity at the highest possible frequency which assures that no ghosting will appear. By band-limiting our texture function only in the direction of disparity we assured that all possible texture values for L contribute to the same texel. This way we effectively avoided ghosting, since the correct texture values always contribute to the corresponding output pixels. As we take the current view-point into account, the closer the virtual camera is to one of the input cameras, the fewer frequencies are cut off from that image and the output image will contain more details. If the input camera and virtual camera coincide, all detail is preserved. Note that as the size of our filter is based on the geometric uncertainty and position of the input cameras, we implicitly take the sampling density into account.

Since the support of the applied low-pass filter can theoretically become arbitrarily large, we take two simple steps to alleviate the needed effort. First, we make strong use of GPU processing power. The whole filtering algorithm is implemented as a pair of vertex and fragment shaders. Second, we trade off detail for speed by applying a multi-resolution technique. We set a threshold ν for the filtersize μ in texture space. If this threshold is exceeded we use the n-th level of the input image-pyramid computed in a preprocess, instead of the image itself, with $n = \log_2(\frac{\mu}{\nu})$. Note that this approach has almost no effect on the visual quality of the output, since a large filtersize implicates a small weighting factor for an input camera and

therefore only a small contribution to the output image, but may speed up the whole rendering process by a factor of roughly 3.

Interestingly, depending on the movement, the constant change in blur in the output image can evoke the impression of repeatedly changing speed, even if a movement is in fact constant. We can solve this problem by applying a simple motion blur technique. If the viewpoint does not change, the image quickly converges to the optimal solution.

4.2 Floating Textures

In the following we describe our second approach, called floating textures. The plenoptic function $\mathbf{P}(\theta, \phi, \lambda, t, x, y, z)$ describes the flow of light as a 7D function for every viewpoint (x, y, z), viewing direction (θ, ϕ), point in time t and wavelength λ [AB91]. Most IBR systems deal with a 5D subset of this function, discarding time and wavelengths. The goal of every IBR sytem is to reconstruct this function as good as it gets.

For simplicity of the analysis we will assume an occlusion free scene. In praxis the occlusion problem can be handled by establishing a visibility map, as described in [CTMS03]. Let us think of any geometry we want to display as a function $\mathbf{G} : (x, y, z, \theta, \phi) \rightarrow (x_o, y_o, z_o)$ which describes how viewing rays are mapped to 3D coordinates on the objects surface. $\mathbf{G}$ is only defined for rays hitting the object, but this is not crucial, since we simply discard the computation for all other viewing rays. Let $\mathbf{G_O}$ be the geometric function of the original object we want to display and $\mathbf{G_A}$ be the function for the approximated object, aquired from the input images. Given that we know the projection mapping $\mathbf{P_k} : (x, y, z) \rightarrow (s, t)$ which describes how three-dimensional points are mapped to pixel indices in the k-th image, these can be used to derive the corresponding color values in the input images, given by the image function $\mathbf{I_k} : (s, t) \rightarrow (r, g, b)$. Then, any weighted linear interpolation scheme $\mathbf{A}$, as used in almost all IBR systems, can be formulated as

$$\mathbf{A}(x, y, z, \theta, \phi) = \sum_k \mathbf{I_k}(\mathbf{P_k}(\mathbf{G_A}(x, y, z, \theta, \phi)))w_k(x, y, z, \theta, \phi) \qquad (4.1)$$

with x, y, z being the current viewing position, θ, ϕ being the viewing direction and w_k is the weighting function of the k-th camera, with $\sum_k w_k(x, y, z, \theta, \phi) = 1$.For three reasons these schemes can almost never reconstruct the correct values of the plenoptic function.

1. $\mathbf{G_O} \neq \mathbf{G_A}$ in most cases. Therefore the input values for the projection mapping are already incorrect.

2. Due to calibration errors, $\mathbf{P_k}$ is only an approximation and may result in the wrong pixel indices.

3. Equation (4.1) tries to linearize the plenoptic function. But $\mathbf{P}$ is not necessarily in the solution space of $\mathbf{A}$ for all input parameters.

Therefore no correct solution can be found in most cases.

In our floating textures approach we solve this problem in the following way. Given our premises, all needed information to synthesize the correct solution is given by the input images. But, knowledge about how to combine them is missing. We can however assume that corresponding pixels have a set of similar properties, like color or gradient constancy. Trying to solve this problem leads us to optical flow techniques, which have proven to work well in finding dense correspondences between two images. However, if correspondences have to be established over large spatial distances, optical flow techniques fail. We therefore propose a symbiosis of linear interpolation and optical flow techniques in our floating textures approach, which will be described in the following. The algorithm is also summarized in Fig. 4.2.

First three images I_0, I_1, I_2 of the scene are rendered from the current view point using the three nearest input images and the weighting functions w_0, w_1 and w_2 are established, as described in Sect. 4.3. Each of these images is in case a very coarse approximation to the plenoptic function at the current viewpoint.

Using these as a starting point, the mentioned problems of the optical flow are circumvented and we can use it to estimate the pairwise flow fields $\mathbf{W}_{01}, \mathbf{W}_{10}, \mathbf{W}_{02}, \mathbf{W}_{20}, \mathbf{W}_{12}$ and $\mathbf{W}_{21}$ between each two of the three input images with sub pixel accuracy. In the next step the flow fields are linearily combined, based on the weighting functions and applied to I_0, I_1, I_2 respectively to acquire the warped versions of the images. Blending these three images finally reveals the resulting image I_{out}. The process is summarized in the following function.

$$\begin{aligned} I_{out} \quad = \quad & ((w_1\mathbf{W}_{01} + w_2\mathbf{W}_{02}) \circ I_0)w_0 + \\ & ((w_0\mathbf{W}_{10} + w_2\mathbf{W}_{12}) \circ I_1)w_1 + \\ & ((w_0\mathbf{W}_{20} + w_1\mathbf{W}_{21}) \circ I_2)w_2 \end{aligned} \tag{4.2}$$

$(\mathbf{W} \circ I)$ warps the image I according to the flow field $\mathbf{W}$.

Note that our floating textures do not fulfill the epipolar constraint anymore, but this does not degrade visual quality. In addition, the epipolar constraint assumes perfectly calibrated cameras, which can not always be provided, but our floating textures can also handle these cases reliably.

4.3 Choice of Input Cameras

Since the resampling density and performance of the optical flow is directly correlated to the choice of input cameras, it is crucial to use only those cameras in the image synthesis step which are closest to the new viewpoint,

depending on the chosen distance measure. These will contribute the highest frequencies, and the most detail, after resampling, as well as the smallest differences on the objects surface when rendered.

To prevent sudden "texture jumps" and preserve view-dependent effects we base our choice of cameras on a hybrid of the approaches by Debevec *et al.*[DTM96] and Pulli *et al.*[PCD+97]. The viewing directions of every input camera are projected onto a unit sphere surrounding the object. From this set of points we compute the Delaunay triangulation. We then choose the three cameras corresponding to the vertices of the Delaunay triangle containing the current viewing direction, and weigh them proportional to the barycentric coordinates. This serves several beneficial purposes. First, the current viewing position will always lie inside the beam spanned by the surface point and the three camera vertices. Therefore view-dependent effects are well captured. Cameras outside the beam are not considered for color computation, which allows for an almost completely unstructured camera setup. The only condition we set is that the object has to be completely visible in the input images. And Finally, choosing nearby cameras also minimizes the amount of ghosting and therefore reduces the effort of band-limiting the texture functions, as well as minimizing the error due to unconsidered occlusion in small scale features.

If used for light field rendering or if very different field of views are used for the input and rendering camera, we can slightly change this approach, if needed. Considering a two-plane parameterized light field, we triangulate the camera plane in a preprocess. For any given viewing ray, which intersects the camera plane at position (u', v'), let the closest samples be $(u, v), (u+1, v), (u, v+1)$, or $(u+1, v), (u+1, v+1), (u, v+1)$, depending on the triangulation and intersection point. We now simply derive the corresponding color values and allocate each of them to one of our three images I_0, I_1 or I_2. If done consistently we created the same basis for our floating textures algorithm, as when using the simpler, but faster method. The filtered blending approach naturally extends to more than three cameras.

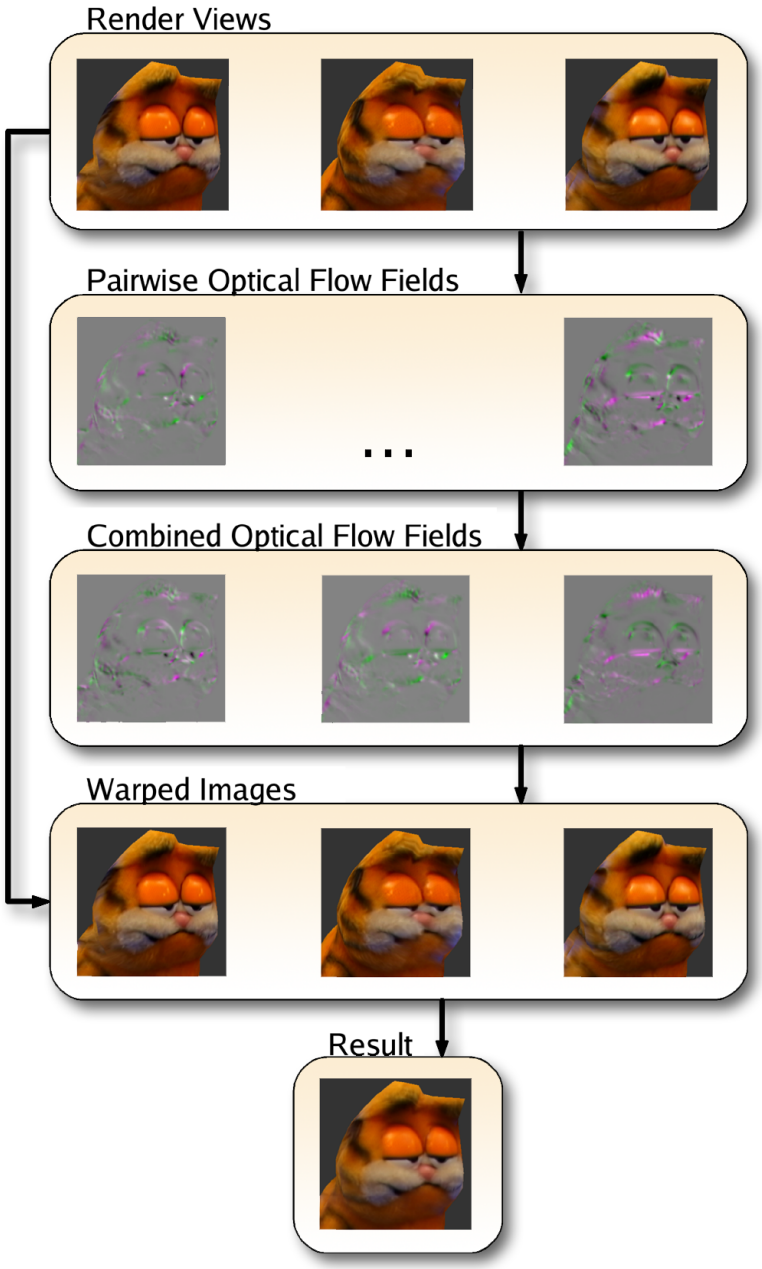


Figure 4.2: Floating textures: Rendering the proxy from the output viewpoint, we separately project three input images onto the geometry proxy. We compute the optical flow field between each pair of rendered output images. We warp the images according to the respective optical flow fields and recombine the warped images to obtain the result image.

Chapter 5

Implementation

We have implemented both proposed algorithms on an NVidia GeForce 7800 graphics card using OpenGL/GLSL. For the filtered blending approach we add/subtract the geometry uncertainty offset from the vertex position. Reprojecting the new positions into the different input images yields the texture coordinates.

For the resampling process during *filtered blending*, we implemented the Mitchell-Netravali cubic B-spline filter as a fragment shader program [MN88]. We compared different filters, e.g., also truncated Gaussian and box filter, and found that the Mitchell-Netravali filter yields the visually most convincing rendering results.

To compute the optical flow in the *floating textures* approach, we rely on a multi-scale implementation of the well known optical flow technique by Horn and Schunck [HS81]. To achieve real-time frame rates the complete algorithm is implemented on the GPU. First, the three nearest photographs are projectively texture-mapped onto the geometry proxy and rendered from the current viewpoint into texture memory. Between these three projection images, the optical flow is computed pairwise. To minimize the local energy, we apply multi-pass rendering. Experiments with our test scenes indicate that the optical flow algorithm converges after 9 iterations per level. In a last rendering pass, the optical flow fields are used to warp the three projection images and linearily blend them together.

Chapter 6

Results, Discussion

Our test data sets include one real-world object, *Garfield*, one synthetically created 3D object, *Bunny*, and the two well-known light fields *Buddha* and *Dragon*. Fig. 1.1 shows examples of the test data sets. Additional data is listed in Table 1.1. To evaluate rendering quality, we compare our rendering strategies to direct (quadra-) linear interpolation as well as to pre-processed band-limited filtering. For pre-processed band-limited filtering, the filter support is set to the smallest possible value to prevent ghosting. In projective texture mapping, we always select the three nearest cameras for interpolation as described in Sect. 4.3.

Our first test scene *Bunny* consists of 49 images rendered from randomly selected viewing directions. The upper three rows in Fig. 7.1 depict the results obtained by the different rendering approaches. When comparing the two leftmost images, which correspond to standard linear blending (Fig. 7.1a) and pre-processed band-limited filtering (Fig. 7.1b), to the two rightmost images, representing our proposed techniques, note how the discontinuities of the checkerboard texture are much better preserved by filtered blending (Fig. 7.1c) and floating textures (Fig. 7.1d). We achieve 100 fps when using our filtered blending approach and 30 fps when using floating textures.

To acquire the calibrated images for the *Garfield* test scene a computer-controlled turntable and a digital camera with a lever arm was used to record 24 images in hemispherical configuration. Rendering results are shown in Fig. 7.1. Again, the noticeable ghosting artifacts along the rim of the eye and the pupil in the images on the left are eliminated in the images on the right. The *Garfield* model is rendered at 108 fps using filtered blending and approximately 30 fps using floating textures.

We also tested our "ghost-busting" approaches for light field rendering using the *Buddha* and *Dragon* data sets (Fig. 1.1). Rendering results are shown in Fig. 7.2. For better rendering quality assessment, some of the details are enlarged. Notice how both proposed approaches are able to prevent

ghosting, Fig. 7.2(c) and (d), while ghosting artifacts are obvious if standard quadralinear interpolation is employed, Fig. 7.2(a). At the same time, much finer detail is preserved than if pre-processed band-limited filtering is used, Fig. 7.2(b). In conjunction with light field rendering, we achieve 45 fps for filtered blending and 31 fps using floating textures.

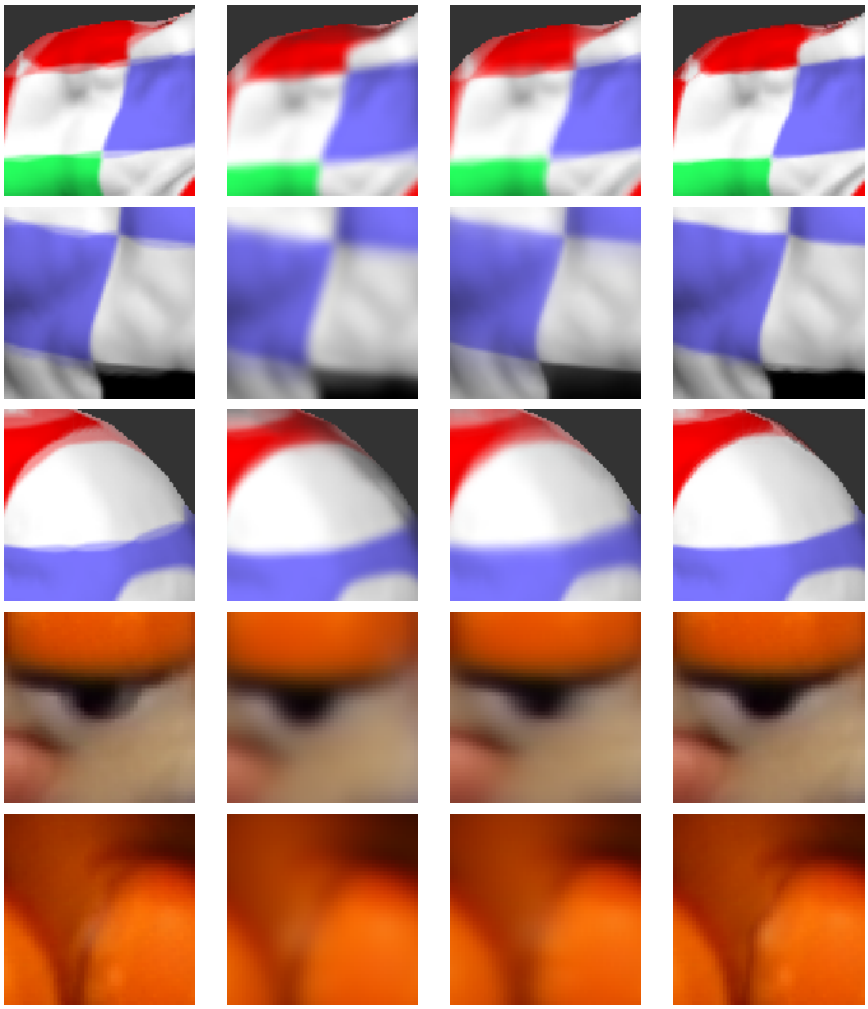
Chapter 7

Conclusions

We have presented two different approaches to achieve ghosting-free rendering results from subcritically sampled light fields, projective texture mapping with approximated geometry, or if texture images are imprecisely calibrated and/or registered. Either method efficiently eliminates ghosting and preserves texture details considerably better than previous approaches. Both techniques are implemented on standard GPU and achieve real-time rendering performance.

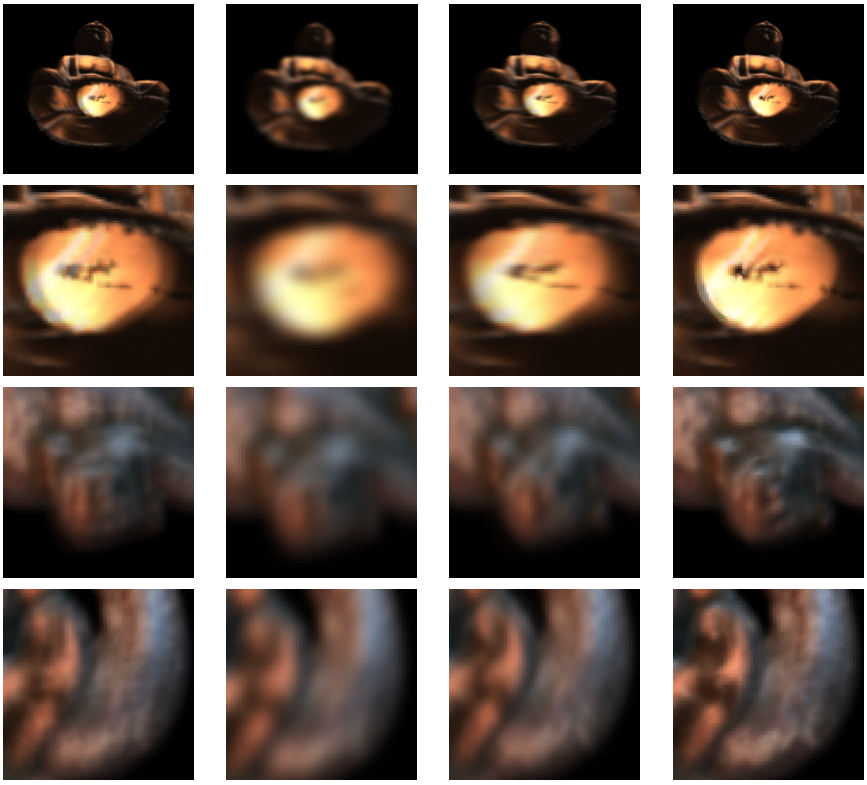
The choice which method to use depends on the number of images, geometry accuracy, and calibration precision. If only very few images are available, the floating texture approach can degenerate to a linear interpolation, in which case filtered blending is the better choice. On the other hand, if projected texture mismatch does not exceed a handful of pixels, floating textures are able to preserve all image detail.

Either approach can be easily adapted to various different image-based rendering scenarios. Filtered blending and floating textures both greatly ease the constraints of image-based rendering: less accurate camera calibration, coarser 3D geometry, and fewer input images are sufficient to still achieve convincing rendering results. With this useful generalization, image-based rendering will hopefully find many new practical applications.



(a) (b) (c) (d)

Figure 7.1: Projective texture mapping of the *Stanford Bunny* (rows 1-3) and of the real-world data set *Garfield* (row 4 and 5): (a) Linear interpolation reveals strong ghosting around high frequency details. (b) Band-limited reconstruction removes ghosting, but the result is excessively blurred. (c) Our filtered blending approach preserves discontinuities considerably better and completely removes aliasing. (d) Floating textures yield optimally sharp results with ghosting completely removed.



(a) (b) (c) (d)

Figure 7.2: Comparison for the two sub-critically sampled light fields *Buddha* (top rows) and *Dragon* (bottom rows): (a) Quadralinear interpolation cannot suppress ghosting artifacts. (b) Band-limiting the entire light field leads to excessively blurry results. (c) Our filtered blending approach preserves most of the details while ghosting is completely avoided. (d) Full detail is preserved if the floating textures approach is used.

Bibliography

[AB91] E. H. Adelson and J. R. Bergen. The Plenoptic Function and the Elements of Early Vision. *M. Landy and J. A. Movshon, (eds) Computational Models of Visual Processing*, pages 3–20, 1991.

[Bau02] Adam Baumberg. Blending images for texturing 3d models. In Paul L. Rosin and A. David Marshall, editors, *Proceedings of the British Machine Vision Conference*, 2002.

[BBM+01] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured Lumigraph Rendering. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432, 2001.

[BMR01] Fausto Bernardini, Ioana M. Martin, and Holly Rushmeier. High-quality texture reconstruction from multiple scans. *IEEE Transactions on Visualization and Computer Graphics*, 7(4):318–332, 2001.

[CCST00] Jin-Xiang Chai, Shing-Chow Chan, Heung-Yeung Shum, and Xin Tong. Plenoptic Sampling. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 307–318, 2000.

[CTMS03] J. Carranza, C. Theobalt, M. Magnor, and H. P. Seidel. Free-viewpoint video of human actors. In *SIGGRAPH '03: Proceedings of the 30th annual conference on Computer graphics and interactive techniques*, pages 569–577, 2003.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20, 1996.

[DYB98] Paul Debevec, Yizhou Yu, and George Boshokov. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. Technical report, 1998.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The Lumigraph. In *SIGGRAPH '00: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, 1996.

[HS81] B.K.P. Horn and B.G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 16(1–3):185–203, August 1981.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2 edition, 2003.

[IMG00] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically Reparameterized Light Fields. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 297–306, 2000.

[LCM+06] Yang Liu, George Chen, Nelson Max, Christian Hofsetz, and Peter McGuiness. Undersampled Light Field Rendering by a Plane Sweep. *Computer Graphics Forum*, 25(2):225–236, June 2006.

[LH96] Marc Levoy and Pat Hanrahan. Light Field Rendering. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996.

[LK81] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. Seventh International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[LKG+03] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatial appearance and geometric detail. *ACM Trans. Graph.*, 22(2):234–257, 2003.

[LS04] Zhouchen Lin and Heung-Yeung Shum. A Geometric Analysis of Light Field Rendering. *International Journal of Computer Vision*, 58(2):121–138, 2004.

[MN88] Don P. Mitchell and Arun N. Netravali. Reconstruction Filters in Computer-Graphics. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 221–228, 1988.

[MP04] W. Matusik and H. Pfister. 3D TV: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *SIGGRAPH '04: Proceedings of the 31st annual conference on Computer graphics and interactive techniques*, pages 814–824, 2004.

[PCD+97] Kari Pulli, Michael Cohen, Tom Duchamp, Hugues Hoppe, Linda Shapiro, and Werner Stuetzle. View-Based Rendering: Visualizing Real Objects from Scanned Range and Color Data. In Julie Dorsey and Phillipp Slusallek, editors, *Proceedings of the Eurographics Workshop on Rendering*, pages 23–34, 1997.

[RCMS99] Claudio Rocchini, Paolo Cignomi, Claudio Montani, and Roberto Scopigno. Multiple textures stitching and blending on 3D objects. In *Proceedings of the Eurographics Workshop on Rendering*, pages 119–130, 1999.

[SSS06] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH '06: Proceedings of the 33rd annual conference on Computer graphics and interactive techniques*, pages 835–846, 2006.

[SYGM03] J. Stewart, J. Yu, S. J. Gortler, and L. McMillan. A New Reconstruction Filter for Undersampled Light Fields. In *Proceedings of the Eurographics Workshop on Rendering*, pages 150–156, 2003.

[VBK05] S. Vedula, S. Baker, and T. Kanade. Image based spatio-temporal modeling and view interpolation of dynamic events. *ACM Transactions on Graphics*, 24(2):240–261, April 2005.

[WAA+00] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 287–296, 2000.

[ZKU+04] C. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. In *SIGGRAPH '04: Proceedings of the 31st annual conference on Computer graphics and interactive techniques*, pages 600–608, 2004.

[ZWT+05] Kun Zhou, Xi Wang, Yiying Tong, Mathieu Desbrun, Baining Guo, and Heung-Yeung Shum. Texturemontage: Seamless texturing of arbitrary surfaces from multiple images. In *SIGGRAPH '05: Proceedings of the 32nd annual conference on*

Computer graphics and interactive techniques, pages 1148–1155, 2005.